Control Theorists are from Mars
Computer Scientists from Venus

Romain Jobredeaux, Alwyn Goodloe, Eric Feron
Georgia Tech and NASA Langley Research Center
Take Home Message

• Major Gap/Lack of communication between Computer Scientists and Control Theorists
• Impediment to progress (and profit)
• A small community is working on it
Motivation

- Sixty years ago control theory and computing were both very young and the scientists and engineers that pioneered each of these fields knew about the other.
- Control Theory and Computer Science have evolved into mature disciplines.
- As a consequence they are barely able to communicate with each other.
- It is increasingly recognized that this disconnect is an impediment to progress.
WHY THE DISCONNECT?
Barriers to Communication: Intuition

Physics Based
• Control a physical system
  – Described by system of Differential Eqs
  – It’s about the physics
• Data: sensors and filters
• Frequency domain analyses

Computation Based
• Computation
  – Computer typically the only physical system of concern
    • Heap, treads, etc.
• Data structures + algorithms + communication
• Hard-Real-time scheduling
Barriers to Communication
Applied Math

Controls

• Ordinary Differential Eq
  – Stability theory
  – Transform Theory
• Linear algebra
  – Multi-linear algebra
• Integral equations
• Real analysis
  – Measure, integration, Hilbert spaces, etc.
• Functional analysis
• Lie algebra

CS

• Logic
  – Temporal
  – Higher-Order
  – Proof theory
• Graph theory
• Semi-rings
• Lattice theory
• Combinatorics
• Universal Algebra
• Category Theory
Same Words Different Meaning

• Both fields often use the same words differently
  – Sometimes with varying degrees of precision
• For instance
  – Control
    • Control flow vs control of physical system
  – State vs. state variables
  – Deterministic/nondeterministic
  – Reachability analysis
• Many “collaborative” efforts never get past this terminology barrier
Stereotypes

• CS = coding
  – Lacking foundations
• Hasn’t real-time workshop eliminated this stuff?

• CT think computer really is whatever the Simulink model says
• All that data just magically moves around the system
• Correctness = Stability
State of the Practice I

• Control theorists work closely with designers of the physical system
  – The models work
• Models and simulations are employed to get the control system right
  – Non-trivial in an aircraft as each “controller” has many many modes
• Gains adjusted through testing
State of the Practice II

• Programmers writing the control code are usually mechanical or aero engineers that learn almost all their programming on the job
  – Control loops generally straightforward
  – Little knowledge of theoretical/formal aspects of CS

• Computing experts employed to “integrate” the different applications
  – Moving data to the right place at right time
  – RTOS/Scheduling
  – Usually don’t get into the details of the “applications”
Observations from the Field

• Two co-authors observations from visiting a large airframe manufacturer
  – Control design department is a distinct entity from the software implementation and analysis department
  – Control designers send Simulink-like diagrams (Scade) to software team
  – Communication among groups is limited
  – Separation reinforces stereotypes
But It Works

• This separation of Controls and Computing communities has evolved over decades
• Many in academia, industry, and government seem happy to maintain the status quo
• After all things work
• Why change?
WHY IT IS AN IMPEDIMENT TO PROGRESS
Challenges

• Challenge to build tools to verify sophisticated systems
  – More than simulate

• Model based design promises to speed design to implementation
  – Many outstanding issues when used in safety-critical systems

• Securing control systems
Advanced Control Theory

- PID controllers well understood and trusted
- More sophisticated controllers have distinct advantages for safety-critical systems
- Robust controls
- Adaptive controls
- Verification?
Predictability Sacrificed

- Safety and predictability not the same
- Historically regulations seek required levels of safety through predictability
- It will take more than a mathematical argument to convince regulators to move away from something that has worked for fifty years
- The software, controls, and safety communities all need to work together to make the case that we know how to assure these
Certification Costs

• 50% of project development costs for software development and certification at Boeing
• These $Billions will be an issue for small UAV based startups as UAV get introduced into the NAS.
Simulation

• Engineers today use simulation in many cases where they once used experiment or analysis

• The simulations can be used in this role because:
  – The simulation executed a model that encodes knowledge about the physical world that has been validated over time
  – Efficacy of simulation rests heavily on the continuous nature of the mathematics used
Model Based Design (MBD)

• Simulink and Scade very popular with industry
• But we must do better than “verification by simulation”
• Computer scientists know that in absence of a published mathematical syntax and semantics, it isn’t really possible to reason about models constructed in any language
• Simulation results may not hold when run on “real systems”
MBD Challenges

• Control theorists should be able to reason about their models using analysis, algebra, and logic
• Fault models should be easily expressible and to reason about at all levels of system design
• Need tool support for a rigorous theory of refinement and autocoding
  – Boxes and lines to contracts, threads, and communications
  – Including the scheduling theory that reflects how the tasks get executed
Security

• Security: In spite of what your models say it’s the implementation attackers go after
• Control theorist need to think about what attackers can do to fool their control systems
  – What are the right attacker models
  – What kind of systems are robust to attack
    • Maybe robust and adaptive controllers are MORE vulnerable to attack (we don’t know)
WHAT CAN BE DONE ABOUT IT?
Do I Need TWO PhDs

• Not if we are going to make progress
• So what is the minimum you need to know to make a contribution?
• Let's start from what both sides need to know to start a conversation
  – I challenge each side to try to use the other's terminology as much as possible
  – Figure out ways to precisely distinguish different
• You will need to know more to research or work in a particular area
Basic Controls for CS

• First 100 pages of Susskind’s “Theoretical Minimum” probably gets you enough physics to get by
• Frequency domain vs Time domain
  – Classic vs modern control theory
• Basics of transfer functions
• Basics of PID controllers
• What is stability
• Review linear algebra
  – Eigenvalues and block matrices
• Read a block diagram
Basic CS for Controls

- Basic programming
  - Data structures beyond a matrix
- Logic as applied to CS
- Basic automata theory
- Basics of embedded processors
- Basics of embedded networks
- Multitasking
- Real-time scheduling
Examples

• Building a provably verified optimization tool a NASA computer scientist learn a lot about optimization and controls, the control theorist just acted as a “domain expert”
• Building a tool to verify hybrid systems requires the computer scientist learn some stability and the control theorist needs to learn about automata, bisimulation, etc.
DO-178C Verification Objectives (Level A)

**System Requirements**
- Accuracy and Consistency
- Compatibility with target computer
- Verifiability
- Conformance to standards
- Algorithm accuracy

**High-Level Requirements**
- Consistency
- Compatibility with target computer
- Verifiability
- Conformance to standards
- Partitioning Integrity

**Low-Level Requirements**
- Compliance
- Traceability
- Compliance
- Traceability
- Compliance
- Traceability
- Compliance
- Robustness

**Software Architecture**
- Compliance
- Traceability
- Compliance
- Traceability
- Completeness and Correctness
- Compatibility with target computer

**Source Code**
- Verifiability
- Conformance to standards
- Accuracy and Consistency

**Executable Object Code**
- Compatibility with target computer

**Autocoding**
- Abstract Interpretation
- GCC / Compcert

**Development Activity**

**Review/Analysis Activity**

**Test Activity**

RTCA DOCUMENTS (CIRCA 2011):
- DO-333: Formal Methods
- DO-331: Model-Based Design
Credible Autocoding Process

Model

Control Semantics: stability, bounded-ness, transient performances, stability margins, etc.

Axiomatic semantics (ellipsoids) & operational semantics (plant model)

C Code

Binary

Proof Verification Backend

Static Analyzer
Credible Autocoder
Certified Compiler
Control analysis results in...

Control analysis yields an ellipsoidal invariant set

- Versatile i.e. encode both stability and performance measures
- Efficient computationally: interior point method to solve the appropriate linear matrix inequality
- Use iqc-beta in practice (megretski/rantzer)
MODEL LANGUAGE (SUBSET OF SIMULINK)

1. Allowed simulink blocks: delays, gains, sums, input, output,..

2. Basics blocks: well-defined semantics..

3. Composite blocks: vague semantics

4. Insert ellipsoid invariants sets from analysis: annotated model
Hoare Logic, pre-/post-conditions

% {1\textsuperscript{st} order precondition on state}
LOC 1
% {1\textsuperscript{st} order postcondition on state}

% {1\textsuperscript{st} order postcondition on state}
\rightarrow
% {1\textsuperscript{st} order precondition on state}

% {1\textsuperscript{st} order precondition on state}
LOC 2
% {1\textsuperscript{st} order postcondition on state}

% {1\textsuperscript{st} order precondition on state}
LOC 1
LOC 2
% {1\textsuperscript{st} order postcondition on state}
ACSL

1. A specification language used to express invariants for C programs.
2. Contract-like structure → hoare triples

PVS

Proof assistants provide:

1. A language to express mathematical properties
2. An interactive mode where user can provide a manual proof of every claim
3. Proof strategies
Verifying the generated code

- Each line of code is annotated with a contract
- Contract describes how the code transforms the invariant set
- For each contract, generate a first order logic property to be proven
- Prove it using a proof assistant (with automation)
Future: Credible Autocoding of real-time optimization

- Real-time Optimization increasingly present in Aerospace applications
  - Complex control tasks (P&W integrated F-135/F35 Model Predictive Control, Control surface Allocation): Fuller
  - Autonomy and path planning (JPL/MIT): Ackimese/Blackmore

- First shot in 2000 with Larry Mc Govern’s PhD thesis (MIT)
- Reappears ~2012 with Stephan Richter (ETH Zurich), Colin Jones (EPFL).
- Goal: Exploit & revise existing optimization autocoder: cvx-gen, but also nonconvex optimizers (Korda)

A New Generation

- A small number of “old dogs” are training a new generation to work in both worlds
- Breaking academic silos not easy as academia rewards specialization rather than breadth
- They are rare, but you can find them if you know where to look
Questions?